Certificate Support in esp8266ndn Library

- esp8266ndn: library for developing NDN app on ESP8266/ESP32 MCU
  - esp8266ndn is used in building sensing deployments at Memphis & NIST.
- esp8266ndn supports HMAC and EC signing and verification, but application must provide uncompressed kit bits.
- This project: add NDN Certificate v2 decoder and encoder.
  - It paves the way of implementing trust schema, ndncert-client, and secure device onboarding in the future.
- Requirement: must have ESP8266 or ESP32 hardware.

Tiny Forwarder with esp8266ndn Library

- esp8266ndn: library for developing NDN app on ESP8266/ESP32 MCU
  - esp8266ndn is used in building sensing deployments at Memphis & NIST.
  - In both systems, sensor nodes are one hop away from NFD.
- This project: create a tiny forwarder on ESP8266/ESP32.
  - We explore a specific design: per-face Bloom filters as PIT, instead of name-based PIT.
  - Sacrifice preciseness of PIT name matching and InterestLifetime, in exchange for lower memory requirement.
- If successful:
  - Building sensing deployments do not need to depend on infrastructure network.
  - Same design can be ported to NDN-RIOT.
- Requirement: must have ESP8266 or ESP32 hardware.

HMAC and Merkle-Hash-Tree Signatures in ndn-cxx

- SignatureHmacWithSha256: hash-based signature with provenance.
  - defined in NDN Packet Format, implemented in CCL, but missing in ndn-cxx.
- SHA-256-Merkle-Hash-Tree: aggregated signing, individually verifiable.
  - defined in CCNxx 0 packet format, but we should explore this algorithm.
  - When producing many Data packets, the producer only needs to perform one RSA/ECDSA signing, while each Data packet is individually verifiable.
- This project: add HMAC and Merkle hash tree signing/verification.
  - HMAC implementation is intended to be mergeable to the codebase.
- Requirement: understand crypto; can write C++14.

- **Junxiao Shi**

NDN File System without FUSE

- NDNFS: an “NDN-friendly file system”. https://github.com/remap/ndnfs-port
  1. Copy files to FUSE <= cannot use existing folder of files.
  2. NDNFS daemon segments the file and stores signatures <= several open bugs.
  3. Data are then served over the network.
- This project: explore a different design.
  1. Program works with an existing folder on the filesystem. No FUSE.
  2. File segmented and Data signed on the fly upon Interest arrival.
  - It’s as simple as `` `python3 -m http.server` ``.
- Requirement: know either ndn-cxx or CCL.

Modernize ndn-js

- ndn-js: JavaScript client library for Named Data Networking.
  - NodeJS: works reasonably well.
  - Browser: written in 2013, not kept up with rapidly evolving JavaScript ecosystem.
- This project: make ndn-js integrate well in a modern JavaScript project.
  - Replace Waf build process with a **bundler** such as Brunch or Webpack.
    - `require("ndn-js")` should work in web application.
    - Dependencies should come from NPM, not `contrib/` folder.
  - Provide **TypeScript declaration** file.
    - This enables type checking of ndn-js APIs in a TypeScript project.
  - Add **Promise**-based APIs as an alternative to callback-based APIs.
- Requirement: can write JavaScript.

Packet03, Data Synchronization, and More in repo-sql

- repo-sql: alternate NDN repository implemented backed by a SQL database.
  - https://github.com/3rd-ndn-hackathon/repo-sql
- This project: a refresh of repo-sql.
  - NDN **Packet Format 0.3**: Selectors are gone.
    - Redesign database schema and query procedure.
    - Reimplement QueryProcessor without selectors.
  - Add some **commands**: let's compete with repo-ng.
    - TCP bulk insertion.
    - Data insertion command.
  - Data **synchronization** using PSync: repo-ng does not have this, but we have.
    - Start with one-way mirroring: secondary instance mirrors a subset of Data from primary instance.
- Requirement: can write C++14 and know ndn-cxx.

- **Junxiao Shi**